EXHIBIT J-1

Transcript Of "Bring Your Android App to Chrome OS - Google I/O 2016" Presentation

May 19, 2016

0:00 LUIS HECTOR CHAVEZ: Hello, everyone.

0:01 Welcome to Google I/O. My name is Luis Hector Chavez

0:04 and I'm one of the engineers in the Chrome OS team.

0:07 Maybe you saw today's earlier announcement

0:09 that we're bringing Android apps to Chrome books.

0:13 Yay.

0:18 So I'm super excited to be here because I can finally

0:20 tell you guys what we've been working on

0:22 for the last few months.

0:23 So it's not a secret anymore.

0:26 All right, so first of all, let's

0:29 talk a little bit about why we're doing this

0:32 and why it should be interesting to you.

0:34 First of all, Chrome OS is a growing platform.

0:37 While overall PC shipments are declining,

0:39 Chrome OS continues to grow.

0:41 In fact, last year we had 32% year over year growth.

0:45 The numbers get even better in certain segments.

0:47 For instance, Chromebooks are the number one device

0:50 in K-12 edu in the US.

0:54 There are millions of active users

0:56 with more than 50 different devices from 13 OEMs in 44

0:58 countries, and we've got lots of amazing Chromebooks coming out.

1:05 OK, so we have a large, growing platform

1:07 with a large number of potentially new users.

1:10 This is where you Android Developers

1:12 come into the picture.

1:14 All Android developers want more users.

1:18 All Chromebook users want more apps,

1:20 but they just don't want any app.

1:22 They want the apps they know and love.

1:26 So we believe that bringing Android apps to Chromebooks

1:28 provides a middle ground that helps both groups of people.

1:32 OK if a developer wanted to bring their application

1:35 in Chromebooks, what are their options?

1:39 Let's start with a non-solution.

1:41 You can write a Chrome OS app.

1:43 This is essentially writing your application twice.

1:45 This requires you to learn a new platform, which

1:49 might yield an application that looks more native,

1:52 but it's a lot of work.

1:53 It's a lot of upkeep, and you need

1:55 to maintain two potentially separate code bases.

1:57 So it's not an option for most people.

2:01 Another option was to build an HTML5 application.

2:05 This might yield an application that looks and feels

2:08 native in Chrome OS, but you would

2:10 need an Android HTML wrapper, which

2:13 doesn't look super native.

2:15 It also doesn't take a lot of advantage of the Android

2:18 platform.

2:19 So it's still not a great option.

2:24 Another option was App Runtime for Chrome.

2:26 We in the Chrome OS team released this

2:28 2 years ago in 2014 as a way to run Android applications

2:32 in Chromebooks.

2:33 We're running full Android instances

2:36 in a Native Client sandbox.

2:38 Unfortunately, it had some challenges for developers.

2:44 Access to some of the system resources

2:45 were restricted due to the advanced nature

2:47 of Native Client sandbox.

2:49 For instance, we had to do a full file system

2:51 emulation because it was not available to us.

2:54 And some apps didn't work great with this.

2:58 Native Client also had a single process execution model,

3:02 which caused some applications to not work great, especially

3:06 the Google Play Store.

3:10 Finally, some of the features that

3:12 are critical for developers, like in-app payments,

3:15 required extra work on your behalf.

3:20 Having said all that, we still believe

3:22 that getting Android applications running

3:23 as a fully integrated native Chrome OS apps

3:26 was the right idea, so we made a lot of improvements

3:30 and we're building a whole new platform to run

3:33 Android apps on Chromebooks.

3:36 All right, so many of you might have

3:39 missed today's earlier demo, so I'm going to give a shorter

3:43 one right now.

3:46 OK, so first of all, this is standard Chrome OS desktop,

3:51 but you can see we have the Play Store now.

3:53 Yay.

3:57 And, of course, I don't have internet

3:59 so there's no Play Store.

4:02 But fortunately I already installed a few apps

4:05 like Gmail.

4:07 Ta-da.

4:08 So one of things we're doing here

4:10 is that we do have multi-window support.

4:21 And not only that, we can also change the size of the window.

4:24 We can also maximize it, stuff like that.

4:28 Neat things.

4:34 We also support integration with some Chrome OS

4:37 native notifications-- for instance, hello, world!

4:50 And, of course, as I mentioned before, I don't have internet,

4:53 so let's skip that.

4:54 Oh, there you go.

4:56 So as we see, we can have offline access in Android apps

4:59 now.

5:04 Right let's go back to the slides.

5:10 OK, we already saw what it looks like,

5:12 so let's talk a little bit more about how it's implemented.

5:18 First of all, we're not using Native Client anymore.

5:21 We're using a brand new sandboxing mechanism.

5:27 It uses existing Linux namespaces

5:30 to isolate Android and Chrome OS.

5:32 These are you already used in Chrome OS

5:34 and in some of our cloud offerings.

5:36 We're using that mount, process, user, network,

5:39 and IPC namespaces.

5:43 The core security team also developed new features

5:46 for this.

5:47 Since Android is running directly

5:49 on top of the Linux kernel, this increased the attack surface

5:52 a little bit more than we were comfortable with,

5:56 so the Chome OS security team developed a new way to add

6:00 alternate sys call tables.

6:02 This is more efficient and configurable

6:04 than existing system call filtering techniques

6:06 like seccomp-bpf, which, by the way, the Chrome OS team

6:10 also developed.

6:13 Having both Android and Chrome OS

6:15 do their own compositing would make everything slower,

6:18 so we have a shared compositing model where there is only

6:20 one overall compositor.

6:22 This makes things a lot faster and more responsive.

6:26 Finally, since we know that a lot of developers target

6:31 indicate for arm only, we're providing just-in-time binary

6:34 translation so that their already existing arm

6:37 applications can run on x86 devices like the Google

6:40 Chromebook Pixel with no work required on your part.

6:47 So this new approach still maintains a high level

6:49 of security in both operating systems.

6:52 You get to keep all of Chrome OS' security features

6:54 like verified root, user data encryption, continuous updates,

6:59 and Android still has [INAUDIBLE] Linux running.

7:04 There are no virtual machines or emulation going on,

7:06 so you get full native performance.

7:10 And finally, since we're running the whole Android system,

7:14 you get to run all of Android Marshmallow within Chrome OS.

7:17 This includes the Google Play Store.

7:21 So something that we learned with App Runtime for Chrome

7:25 is that integration matters.

7:26 The more integrated the apps are with the rest of the operating

7:29 system, the better they look and feel.

7:32 So we only grant direct hardware access

7:34 to Android in very few locations.

7:35 The rest of the time, the hardware

7:37 is still managed by Chrome OS.

7:39 But still, we had a lot of integration points

7:41 so that Android developers can still call the same APIs,

7:45 and those APIs underneath will make calls to Chrome OS.

7:49 This allows both operating systems

7:51 to share the same view of the system

7:52 instead of having split views.

7:56 OK, so what does it all translate into,

7:58 and how does it impact your applications?

8:03 Because we're running a full Android system,

8:05 it means that most applications can

8:06 be running Chrome OS without any code changes at all.

8:09 Of course, since this is a new platform,

8:12 you might want to still make some optimizations

8:14 and small tweaks.

8:18 But your app works like a Chrome OS app.

8:23 Most of the things you expect the Android system to do

8:25 will be now provided by Chrome OS instead of being constrained

8:29 to a little window.

8:31 Every task runs in its own window.

8:34 You get Android notifications together with Chrome OS'.

8:37 There's just one App Launcher in shelf.

8:40 Sign-in information will be shared

8:41 between operating systems, so you don't have to log in twice.

8:46 You're also able to share files in the Downloads folder,

8:48 as well.

8:52 Hardware is also integrated so you still get connectivity

8:55 to Wi-Fi and Bluetooth.

8:57 Camera, microphone, audio, and video still work.

9:04 All of Chrome OS' input mechanisms

9:06 are still plumbed through the application,

9:08 so keyboard and trackpad events will

9:10 be delivered to your application,

9:12 and also touch events if the Chromebook supports it.

9:17 Meanwhile, your application still

9:19 behaves like an Android application.

9:22 Since we're running a full Android stack,

9:24 the whole Marshmallow API is available to you.

9:27 This includes Google Play services,

9:29 so things like in-app payments will

9:30 be available in consumer Chromebooks.

9:33 We're also not adding any new APIs,

9:35 so there's nothing for you to learn this time.

9:40 All system services are running, and all the interactions

9:43 between the system services and your application

9:45 are exactly like you would expect.

9:47 This includes things like the application lifecycle events.

9:53 Finally, the hardware is abstracted so that you don't

9:55 need to worry about anything.

10:00 And I'm going to be clear-- while apps

10:02 can be run in Chrome OS, we still

10:05 require Chrome OS users to opt in to use this feature.

10:08 We understand that there are some scenarios in which people

10:10 don't want to enable this just yet.

10:13 We're also providing enterprise and education users

10:16 with additional policies that can enforce if and which

10:19 applications can run on the devices they manage.

10:23 All right, when is this going to be available?

10:25 For you developers, we're enabling this feature on Chrome

10:29 OS 53.

10:30 This should be available in the DEV channel in June,

10:34 and 53 should be generally available

10:37 for all users in the stable channel in September,

10:39 so you still have some time to prepare.

10:43 We'll start by supporting three devices-- the Asus

10:45 10-inch Flip, the Acer Chromebook R11,

10:48 and the Chromebook Pixel 2015.

10:51 We'll gradually bring support to more devices

10:52 in later milestones, so stay tuned.

10:54 And that's the URL where you can check

10:56 which devices will be getting support on which releases.

11:01 So stay tuned.

11:04 OK, it's great that most applications will work out

11:06 of the box without any node changes,

11:08 but let's take a look at the best practices

11:13 you can take into account so your application works

11:17 even better on Chromebooks.

11:22 There's a wide variety of Chromebook form factors.

11:25 All of them have a physical keyboard and trackpad.

11:28 Most of the advices we'll be rolling out

11:30 support to have a touch screen, as well.

11:32 Some of them are even convertible so they

11:34 can switch between a laptop and a tablet form factor.

11:38 While we're testing applications, most of them,

11:41 we found out that they already work pretty well

11:43 with keyboard and mouse.

11:45 Still, not all of them did.

11:47 So it's better that you developers

11:51 take into account testing with a keyboard and mouse.

11:55 Also, since there's essentially new hardware available to you,

11:58 might as well make the best of it.

12:02 So productivity applications can have

12:04 things like hotkey support.

12:08 That'll make users even more productive.

12:11 Some games even lend themselves to be controlled

12:13 using a keyboard and mouse.

12:26 OK, now for the important coding part.

12:29 By default, all Android applications

12:31 set the touchscreen hardware feature to be required.

12:35 Now, I mentioned before that not all Chromebooks

12:38 will provide touchscreen.

12:40 So it's very important that on your AndroidManifest.xml

12:45 you set the android.hardware.touchscreen

12:48 feature to be not required.

12:51 This will allow your application to appear

12:52 in the Play Store for all Chromebooks

12:54 instead of just for the Chromebooks that support touch.

12:57 If you go out of this presentation with just one

12:59 thing in mind, let it be this.

13:04 All right, best practices for multi-windows.

13:07 On laptops, everybody uses multiple windows

13:09 for productivity given that there are more pixels available

13:11 on the screen.

13:14 Now, I did mention a couple of times

13:15 before that we're supporting Marshmallow API,

13:19 and Marshmallow doesn't have the APIs for multi-window.

13:22 I also mentioned that we are not adding any new APIs,

13:25 so how does it work?

13:28 In order to minimize the amount of work needed for everybody,

13:30 we are only supporting a few layouts

13:32 that your application already most likely supports.

13:39 First of all, we have a landscape.

13:40 This is the default layout.

13:42 This uses a Nexus 9 aspect ratio and device [INAUDIBLE].

13:50 We also support portrait, which runs just like a Nexus 5.

13:56 We also have maximized windows, which

13:57 take up all available pixels in the screen.

14:03 Finally, for convertibles, we have

14:04 touch me mode, which is more or less the same as maximize,

14:08 but hides the window decorations and the shelf at the bottom,

14:11 so it's similar to Android's immersive mode.

14:16 We added some window controls to toggle between all

14:19 the available layouts.

14:21 This allows both users and developers

14:24 to choose the right balance of information density

14:27 in the sizes and layouts their applications were originally

14:29 designed for.

14:31 Now, some things you need to be aware of.

14:34 Similar to [INAUDIBLE] multi-window implementation,

14:37 we're not changing the application lifecycle model.

14:40 Exactly one application will be on the onResume state,

14:43 and this is the window that's currently focused.

14:45 All the rest of the windows and applications

14:47 are in the onPause state.

14:50 Also, the out-of-memory killer is integrated

14:53 with the rest of the operating system,

14:55 and it takes the z order of the windows

14:57 into account when calculating the final score.

15:00 So applications that have been least recently used

15:04 will be killed first.

15:08 OK, so since changing orientation also

15:14 changes the physical size of your window or of the device,

15:18 you need to declare the correct Android screen

15:20 orientation so that it only uses the ones it supports.

15:27 Now, it's also important that you

15:28 go read this document, the Handle Runtime

15:30 Changes in the official Android documentation.

15:36 There will be some things that you

15:38 don't expect to change whenever changing orientation.

15:42 For instance, the screen width and height

15:44 will change in ways you don't necessarily

15:46 expect because on normal devices,

15:48 they only flip between them, and here

15:50 you can have totally different values.

15:53 Also, the density DPI setting can change

15:57 when you switch orientation.

15:59 Also, very important-- make sure you invalidate any cache

16:02 resources.

16:03 Since once you change orientation,

16:05 you're effectively changing device,

16:08 you need to make sure that you load the correct resources

16:10 after the orientation has changed.

16:13 Now, these two things are still important for Android

16:16 [INAUDIBLE], so might as well do them right now.

16:23 Also, use the correct task affinities.

16:26 As I showed in the demo, it's possible for a task

16:30 to launch another activity in a new task.

16:33 This will render a new window.

16:35 But if your intention was to have a [? modal ?] window,

16:37 you should put that activity in the same task.

16:41 Otherwise, you might get into application faults

16:45 that you weren't expecting.

16:49 Finally, respect the onPaused state.

16:52 While your application might be paused,

16:54 it's still visible, so make sure you're not

16:56 doing any rendering because that might

16:57 look awkward for the user.

16:59 And it will also consume a lot of battery.

17:05 Using backup and restore effectively

17:06 is also very important.

17:08 One of the best features about Chromebooks

17:10 is that users can just throw out their machines, get a new one,

17:14 login, and all their applications and settings

17:17 will be there.

17:18 So it's not totally required to support backup and restore,

17:22 but it's a very good idea and users will love it.

17:25 Make sure you read the official documentation

17:27 in Backup & Restore.

17:30 There are also some scenarios in which

17:33 Chromebooks can be shared amongst a large number

17:35 of users.

17:36 For instance, in education, students

17:38 don't get their own personal Chromebooks.

17:40 They are shared between students.

17:43 So if you're writing applications for education

17:45 purposes, make sure to take this scenario into account

17:48 and be mindful of the local storage you use.

17:54 In general, Chromebooks support a less amount of sensors

17:56 compared to mobile devices.

17:58 For instance, most Android developers

18:01 are used to having a GPS device to get accurate location

18:04 information.

18:06 Chromebooks don't have a GPS device,

18:08 but they can still get [INAUDIBLE]

18:09 location information through the use of Wi-Fi.

18:12 This might be enough to get the best restaurants around you,

18:15 but it might not be enough to just grab a Chromebook,

18:18 put it in your car, expect to have turn-by-turn navigation.

18:24 So the recommendation is to make sure you don't require hardware

18:27 that might not be available.

18:30 Of course, if your application does

18:31 require some piece of hardware that's

18:33 not available on Chromebooks, we'll respect your decision

18:36 and not show that application on the Play Store for Chromebooks.

18:46 There are some other software features

18:47 that are not going to be supported in Chrome OS.

18:52 Since we want users to have the full Chrome OS experience,

18:55 applications that customize some parts of the UI

18:57 will not be available.

18:59 Applications that provide custom input methods, app widgets,

19:03 live wallpapers, or home screens won't be supported.

19:08 Also, Chrome OS is going to be the device manager for Android,

19:11 so applications that declare themselves

19:13 to be the device admins or manage users

19:16 also won't be supported.

19:23 We will be shipping initially with Android Marshmallow,

19:26 but it's still a good idea to focus some efforts in upgrading

19:31 to N eventually because you'll bring

19:33 a lot of features that will improve

19:35 desktop productivity to users.

19:37 For instance, you will have the ability

19:40 to fully resize activities instead

19:41 of being constrained to one of the four layouts

19:43 we support right now.

19:45 You'll also be able to use drag and drop between applications.

19:49 And there's going to be a new mouse cursor API available.

19:54 So in order to learn more about N,

19:55 you can watch these two presentations

19:59 that aired yesterday today here in I/O,

20:02 the What's new in Android and Multi-Window Mode.

20:05 Also, you can consult the official Android N Developer

20:08 Preview documentation.

20:12 OK, if we have any more questions about this,

20:15 you can consult the official documentation

20:17 for Android Apps for Chromebooks.

20:20 You can also ask questions in Stack Overflow or post

20:22 questions on the Android G+ Community.

20:29 Now, before I go, a lot of you tried

20:32 to get to today's earlier announcement

20:35 but couldn't get to it, so we're going

20:37 to redo the demo that we gave earlier today.

20:40 So here's Katie that will present to you the earlier

20:43 demo.

20:50 KATIE: All right, thanks a lot Luis.

20:52 LUIS HECTOR CHAVEZ: Can we switch to the demo?

20:57 KATIE: All right, very good.

20:59 So apologies this morning that we are a little short on space.

21:03 So we thought with these extra few minutes,

21:04 we could go back through our demo flow.

21:08 So unfortunately, it'll be a mild different second

21:12 rendition.

21:13 Hopefully a little bit better.

21:14 But here we've got a Pixel 2.

21:19 It looks like a regular Chromebook.

21:21 Got our Chrome browser here.

21:23 But you obviously now have the Play Store on the shelf here.

21:27 So we'll go ahead and launch that.

21:29 So here we can actually browse through our apps.

21:34 This looks just like the regular Play Store, so let's go ahead

21:36 and install something.

21:38 So I think we decided Bejeweled is a pretty good game.

21:41 Let's search for that, and we'll go ahead and install.

21:50 Oh, and look at that.

21:51 Coincidentally, apparently Kan's wife

21:54 is asking me to make a poster for Kan's child's birthday

21:58 party.

21:59 So being a good employee, let's go ahead

22:02 and reply to Kan's wife-- I'd love to.

22:08 The great thing about the way we've integrated Android

22:12 is we get the nice in-line reply that you'd

22:14 expect on your tablets and phones,

22:15 so I'll go ahead and send that off now.

22:18 Great.

22:19 So let's go ahead and do that.

22:21 I've already installed Photoshop Mix from the Play Store,

22:25 so we'll go ahead and launch that now.

22:26 So that's actually a picture of Kan's kid,

22:29 I think, feeding a pen to Piglet.

22:33 Odd.

22:34 Anyway, I've been told that it would

22:36 be more appropriate to put ice cream there instead.

22:40 So we'll go ahead and try to find some ice cream here.

22:43 So luckily from Adobe stock, they've

22:45 got some great photos of ice cream.

22:47 It's actually very bizarre, like half neapolitan ice

22:50 cream with some pistachio.

22:54 But that looks kind of big, so why don't we

22:56 go ahead-- we can use the touchscreen

22:58 just like we'd expect.

22:59 We can go ahead and shrink our ice cream cone down.

23:03 That's a little bit better than a pen, I suppose.

23:07 One of the problems here, though, is obviously now

23:09 our cone is kind of over her hand,

23:12 so we'll go ahead and cut out that.

23:14 We can zoom in and zoom in a little bit more.

23:17 We can go ahead again, use the touchscreen.

23:20 That was a chunk out we didn't want.

23:21 We'll go ahead [INAUDIBLE] that.

23:26 There we go.

23:30 That Photoshopping live, not my finest work.

23:40 Probably come up in my next performance review,

23:43 but OK, let's go ahead and save that anyway.

23:48 All right, we're saving that.

23:49 That looks great.

23:52 So I've been told that he wants a poster, so let's go ahead

23:56 and open up Word for that.

23:58 What's really nice about this is I can now just go--

24:01 and I've got regular Word here, throw in some photos

24:04 for Leanne's birthday party.

24:06 There we go.

24:07 That looks pretty good.

24:10 So I want to now give this back to Kan's wife.

24:14 So let's go ahead and we can click that we want

24:17 to share this as an attachment.

24:20 Go ahead, open up the Gmail.

24:22 We hear that's a pretty good product.

24:24 And go ahead and send this off to Nan.

24:29 That looks great.

24:30 Let's go ahead and send that off.

24:33 Oh, what do you know?

24:34 Just coincidentally, someone is attacking me on Clash of Clans.

24:38 So again, with how we've integrated the Android

24:43 framework, chat has just worked.

24:45 This is Shahid right here.

24:49 We'll go ahead-- we better go check out our village.

24:53 So Clash of Clans.

24:55 We go ahead and close Photoshop.

24:57 We don't need that anymore.

25:01 Go ahead, get Clash of Clans rolling for us.

25:04 Oh dear.

25:06 You guys have never, ever seen a demo not work on stage.

25:13 We'll go ahead-- we actually have a slight known bug here.

25:18 Any case, though, we'll go ahead and continue on from that.

25:23 Apologies there.

25:24 Another great game other than Clash of Clans,

25:26 which I also like a lot, is Galaxy on Fire.

25:29 Here we're able to get the nice use of a desktop GPU.

25:34 Everything's super smooth.

25:35 I don't know if it looks super nauseating to you

25:38 all in the audience, but anyway, very good performance here

25:41 as well.

25:44 With that, I can also do in-app purchase, as well.

25:47 So we can go direct, we can go into buy an add-on.

25:51 And go ahead and click Buy.

25:53 And now this will be available on my phone--

25:57 payment successful, great-- as well as my tablet, as well.

26:03 So again, this was just a quick view of what we also

26:06 showed this morning.

26:07 We are, of course, very excited to be working with all of you

26:10 to bring your apps to Chromebooks.

26:11 So thank you very much.

26:21 LUIS HECTOR CHAVEZ: All right, that's it from both of us.

26:23 Enjoy the rest of the show.

26:24 If you have any questions, ask them of the Spaces app,

26:27 or we'll be hanging around down the stage.

26:29 Thank you.

26:29 Bye.